A System For Naming Faces And Vertices In An Adaptive Hierarchical Subdivision Surface

Cross Reference to Related Applications

This application is related to U.S. applications entitled Surface Evaluation having serial number 09/116,553 by Stam, filed July 15, 1998, entitled Boolean Operators For Subdivision Surfaces having serial number 09/164,089 by Lee, filed September 30, 1998, and entitled Parameterization Of Subdivision Surfaces having serial number 09/219,613 by Lounsbery, filed December 23, 1998, all incorporated by reference herein.

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention is directed to a system that provides a name for faces and vertices in an hierarchical subdivision surface and, more particularly, provides unique names from which each face of a surface, each vertex of a surface and each edge of a surface can be unambiguously identified.

Description of the Related Art

Many computer applications model surfaces and objects. Since computers have a limited storage capacity, an object cannot be modeled with an infinite number of coordinate points. To overcome this problem, methods are used to approximate the object surface with surface primitives. One method uses a polygonal mesh comprising a set of polygonally bounded faces. Representing objects with curved surfaces using a polygonal mesh requires approximating the surface with a number of faces or polygons. The error between the model and the real object can be reduced by making the polygons arbitrarily small. When an object is represented as a set of such polygonal faces, and the representation is loaded into or out from the modeling application, the faces, as well as the vertices and edges need to be identified. In conventional systems, often the same face or vertex is provided with a different identity or identifier each time the representation is loaded into the application. Additionally, when the representation is provided to a different but compatible modeling application, a face or vertex can receive another identity. This changing of the identity of faces and vertices results in inefficient utilization of resources. Inconsistencies between representations make it difficult or impossible to identify the "same" face or vertex in representations that are copies of each other. Such inconsistency is a serious problem in modeling systems.

When a model of an object needs to be made more accurate over all or in a particular area, the modeling application divides larger polygon faces into smaller faces and thereby adapts to a higher resolution representation. When this occurs, the system needs to create face and vertex identities that are not duplicates of other vertices and faces that already exist in the representation, and theses new names need to persist

when the representation is loaded and used at a later time such as for input/output.

What is needed is a system that defines a unique name for every face at any level in an adaptive hierarchical subdivision surface, such that the name persists across different executions of the code, so that it can be used for I/O. What is also needed are unique names for the vertices and edges of the polygonal faces of the mesh.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a system that uniquely names or identifies every face, vertex and edge in a hierarchical subdivision surface.

It is another object of the present invention to provide names that persist across executions of an application that uses a polygonal face representation of an object.

It is also an object of the present invention to improve the efficiency of systems that represent objects with a polygonal mesh in a hierarchy

The above objects can be attained by a system that assigns a unique name or identifier to a face that includes information identifying the base mesh face from which the face being named is derived, identifying the base mesh vertex from which the face is derived and identifies the path to the face. When the face name is represented as a fixed length number, the name can include identity of the subdivision level of the face. In association with the face name, the vertices and edges of the face can be assigned unique names.

These together with other objects and advantages which will be subsequently apparent, reside in the details of construction and operation as more fully hereinafter described and claimed, reference being had to the

accompanying drawings forming a part hereof, wherein like numerals refer to like parts throughout.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 depicts the name of a face according to the present invention.

Figure 2 illustrates a base mesh, base face numbering and a vertex numbering of a part of the base mesh.

Figure 3 illustrates a first level subdivisions of the base mesh of figure 2.

Figure 4 depicts a second level subdivision of the base mesh of figure 2.

Figure 5 depicts face numbering or naming and origin designation.

Figure 6 depicts a third level subdivision along with face numbering and origin designation.

Figure 7 illustrates vertex numbers or names and edge numbers or names.

Figure 8 illustrates a vertex shared between faces.

Figure 9 is a flowchart of a naming operation.

Figure 10 depicts hardware which can be used in the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention provides a unique naming scheme for naming faces, vertices and edges of adaptive hierarchical subdivision surfaces. This naming scheme produces a name that can be embodied in a field divided data structure such as illustrated in figure 1. A name 10 in accordance with the present invention includes a base face index 12 (or

name) preferable an integer, a vertex index 14 (or name) preferable an integer, a mesh level 16 preferable an integer and a path 18 that identifies how to reach the face or vertex within the mesh starting from the base mesh (level 0). The path is preferably a stack of integers (bit pairs) in the range 0..3, with coarsest at the top of the stack. The base face index 12 identifies the polygon of the base mesh in which the target face resides. The vertex index 14 is an index to the level one face within the base face in which the target face resides. The vertex index requires a predetermined ordering, such as, counter-clockwise, among the adjacent faces and the vertex name or index is the name of the lowest-order adjacent face, concatenated to the ordering of the vertex around that face. The mesh level 16 is the subdivision level of the target face within the hierarchical subdivision surface. The face name with the concatenated fields 12, 14, 16 and 18 noted above is preferably converted into a 64 bit integer for ease of handling and storage. When the integer is converted into a binary number, the field divided data structure noted above, can be used to access the different variables of the face name.

Polygonal meshes can be used to describe the surface of an object being represented in a computer. The accuracy of the surface representation depends of the number of polygons of the mesh representing the surface. The finer the mesh, the more accurate the representation. As a result, graphical representations of surfaces are divided into polygons depending on the accuracy of the surface needed. An arbitrary collection of polygons can be subdivided into finer and finer meshes as needed. Figure 2 depicts a zero level mesh 30 including three polygons, one a five sided polygon 32, one a four sided polygon 34 and a three sided polygon 36. Figures 2-4 depict the subdivision of the zero level polygonal mesh 30 into a level one mesh 38 (figure 3) and a level two mesh 40 (figure 4). Figures 6 and 7 illustrate a subdivision of the mesh down to level three for a portion

of the mesh. Each of the subdivisions is obtained by determining the midpoints of each edge and the midpoints of each face, and connecting the edge midpoints to the face midpoints to make what are called "quads" which is a technique called Catmul-Clark subdivision. At the point that the subdivision goes from level zero to level one, the entire surface is represented by quads. Each of the quads includes several graphical elements: a face, several multidimensional vertices and four edges. These elements can be identified or named in a number of different ways which allows the representation to be stored, retrieved and used by graphical application programs. For example, it is possible to provide a pointer for each element and create a list of pointers. As another example, the elements can be arbitrarily indexed and listed. However, these methods are arbitrary and when the surface is edited, generally the listing needs to be completely re-created from the beginning. When the representation is stored, the mesh must be rebuilt in exactly the same order. The present invention provides a naming scheme that is independent of such arbitrary requirements.

The present invention assumes that a base polygonal mesh or base mesh is provided that includes a base face name or index for each face and a vertex ordering. Figure 2 shows base face indices of 0, 1 and 2 for the 5, 4 and 3 sided polygons, respectively, and a vertex ordering for face 0 of 0, 1, 2, 3 and 4 in a counter clockwise direction, which is the preferred ordering direction for vertices and edges as discussed herein.

In building a face name, the base face name is the base face name of the face in which the target face resides. In the example of figure 5, using the base face names and vertex ordering shown in figure 2, the face 52 has a base face name of 0. In contrast, the face 53 has a base face name or index of 2.

The vertex index is the index of the level one face or the level zero base vertex of the face as seen from the base mesh face. That is, the ordering of that face associated with the vertex, as indexed from the base mesh FACE that the level 1 face is subdivided from. For face 52, the vertex index or index to the level one face within the base mesh face is 0.

The path is defined by quad tree hierarchical indexing conventions. To define the path, an origin and the axes of the face need to be defined for the face. The origin is the closest vertex of the face to the vertex of the vertex index of the target face and the farthest vertex is the opposite vertex of the face. The base vertex is the origin "00" and the farthest vertex is the "11" vertex as shown in figure 5. The subdivision faces are then provided a binary number assigned counter clockwise, such that the path to face 52 is binary 01 or integer 1. As another example, using the numbering and ordering of figures 2 and 5, the face 54 of figure 6 at level 3, would have an origin "00", as shown, which is determined as the vertex closest to the index vertex and a path of binary 11,10 or integer 3,2.

The name is stored as fixed precision integer rather than as a series of binary pairs. This can cause some ambiguity in the interpretation of the path specified by the integer. For example, if the path is being used in a system in which the path field of the integer is as eight bits, the path 2 could be interpreted as 00,00,00,01, such that the face is a level five face. In a system in which the path field is represented by 12 bits, the face would be a level seven face. To remove this ambiguity, the system indicates, in the face name, the level of the face for which a path is being provided, thereby indicating the size of the field being used for the path. In the example of figure 6, the level is 3.

As a result, the path name for the face 54 in figure 6, would be 0,0,3,3,2.

The vertices of a face, as previously noted, are numbered starting with the origin (the vertex closest to the base face vertex) and numbered in a counter clockwise direction. In the example of figure 7, the vertex 56 of face 58 has a vertex number of 2. Edges are numbered in a similar manner starting with the origin and counting clockwise. In the example of figure 7, the edge 60 is edge number 1.

Figure 8 depicts faces 58 and 62 and the face names of theses two faces will be provided as another example. The face name of face 58 is 0,4,3,3,2 or base face index = 0, vertex index = 4, level = 3 and path =3,2. The face name for face 62 is 0,3,3,1,2 or base face index = 0, vertex index = 3, level = 3 and path = 1,2.

Because vertices can be shared by two or more faces, it is possible to have the same vertex with more than one name. This is inefficient and does not provide a unique name for that vertex. To resolve this, the vertex is given the number or name from the face name that has the lowest ordering. As noted above, the face names are manipulated in the form of 64 bit integers, to determine which has the lowest ordering, the integers are compared and the smallest has the lowest ordering. In the case set forth in figure 8, the vertex 64 would be named based on the ordering of the vertices of face 62, which in the example would be vertex 2.

Because edges can be shared by two faces, it is possible to have the same edge with more than one name. To resolve this, an edge is also given the number or name from the face name that has the lowest ordering.

Once a mesh has been subdivided, a routine to produce the face name from a particularly designated face would be performed. This routine 70, as depicted in figure 9, would allow the user or the application using the naming routine to specify 72, a face to be named. The system would then determine 74, the level of the face followed by determining 76,

the path to the face. Next, the vertex index would set 78, followed by setting 80, the base face index. Below is more detailed pseudo-code for such a routine or program. Note that this routine, when determining the path, starts with the face and works up the quad hierarchy.

```
Routine to create a face name from a specific face in the subdivision
surface hierarchy.

Input: a face
Output: an encoding of that face, made up of:

Base_face_index      (integer)
Vertex_index (integer)
Level (integer)
Path (stack of integers in the range 0..3, with coarsest at the top of
          the stack)
---
Path = {}                          // path starts out empty
pathFace = face
Level = levelOf ( face )           // routine to return face's level
if ( Level == 0)
{
  // face must be a base face
  //
  Base_face_index = indexOf ( face )
  Vertex_index = 0                 // can be anything, actually
}
else
{
   while ( levelOf ( pathFace ) > 1 )
   {
          quadFace = pathFace
          // determine what index this face has, as seen by its parent
face
          //
          pathStep = indexFromParent ( quadFace )
          // stack push: adds step to Path, increasing size of Path
          //
          pushStack ( pathStep, Path)
          pathFace = parentFaceOf ( quadFace )
```

```
    }
    // pathFace must be a level-1 face
    //
    Vertex_index = indexFromParent ( pathFace )
    pathFace = parentFaceOf ( pathFace )
    // pathFace must be a level-0 face
    //
    Base_face_index = indexOf ( pathFace )
}
—
```

Once a face name is available, it is often necessary to access the face identified by the name. A routine to access the face name using the face name would be performed. Below is pseudo-code for such a routine or program.

```
Routine to access a face based on a name representation.

Input: a face name, consisting of:
Base_face_index      (integer)
Vertex_index (integer)
Level (integer)
Path (sequence of integers in the range 0..3, with coarsest at the top of
        the stack)
Output: a face in the subdivision surface hierarchy
---
base = baseFace ( Base_face_index )              // base face with that
index
If Level < 1
   result = base;
else
{
   // start with Level-1 subface of base face at that corner
   //
   result = subFace( base, Vertex_index )
   // iterate as far as Level tells us we have to go
   //
   for (i = 1; i < Level; i++)
   {
        // stack pop: removes value at the start of path, shortens path
```

```
                //
                quad = popStack ( path )
                result = subFace ( result, quad)   // subFace(f, i) is a routine
                                                   // that returns the ith subface
                                                   // of face f
        }
    }

    return result
    —
```

The methods described above can be implemented in special-purpose circuitry, general-purpose circuitry (such as programmable microprocessors) operating under the control of program instructions, or in any combination of such apparatus. However, the methods are not limited to any particular circuitry or program configuration, they can find applicability in any computing or processing environment that can be used for the manipulation of meshes used for constructing surfaces of objects.

As shown in figure 10, the invention can be implemented in computer programs executing on programmable circuitry that can include a processor, a storage medium readable by the processor (including volatile or non-volatile memory and/or storage elements), one or more input devices, and one or more output devices. Program code can be applied to data entered using the input device to perform the functions described and to generate the output information. The output information can be applied to the one or more output devices.

A computer system 100 suitable for performing naming operations upon meshes includes a CPU 102, a display 104, a system memory 106, an internal memory device (e.g., hard disk drive) 108, a user input device(s) 110 (e.g., keyboard and mouse), and a removable storage medium 112 (e.g., floppy, tape, or CD-ROM) read by an appropriate drive 114, all coupled by one or more bus lines 116. Code for a program 118

used to create a face name or used to access a face based on the name representation can be stored on removable storage medium 112, and then introduced to computer system 100 through drive 114 to be either temporarily stored in system memory 106 or permanently stored in internal memory device 108. CPU 102 can then use the introduced program 118 to perform operations, including generating and using or generating one or more data structures 120. Program 118 can also be included within another computer graphics program to assist in naming meshes for other operations.

Each program described above can be implemented in a high level procedural or object oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if desired. In any case, the language can be a compiled or interpreted language. Each such program can be stored on a storage medium or device (e.g., DID, CD-ROM, hard disk or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform the procedures described in this document. The system can also be considered to be implemented as a computer-readable storage medium, configured with a computer program and/or a data structure, where the storage medium so configured causes a computer to operate in a specific and predefined manner. The program 118 can also be included within another computer graphics program to assist in naming faces or accessing faces of meshes for other operations. The program and/or data structure can also be transmitted over a network such as the Internet.

The many features and advantages of the invention are apparent from the detailed specification and, thus, it is intended by the appended claims to cover all such features and advantages of the invention which fall within the true spirit and scope of the invention. Further, since

numerous modifications and changes will readily occur to those skilled in the art, it is not desired to limit the invention to the exact construction and operation illustrated and described, and accordingly, all suitable modifications and equivalents may be resorted to, falling within the scope of the invention.